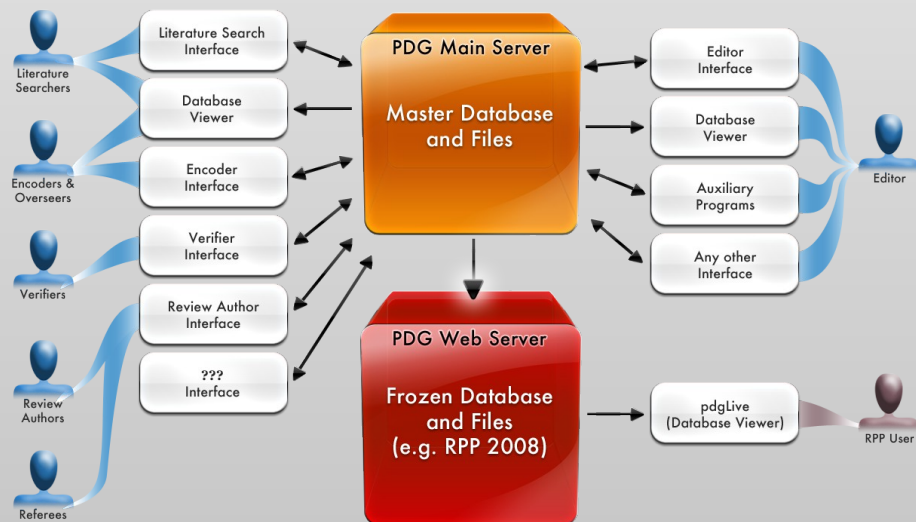


Status of the New PDG Computing System

Juerg Beringer

Physics Division

Lawrence Berkeley National Laboratory



Outline:

- Project status
- **Major success: V0 Release**
- Roadmap towards completion
- Innovative new features
- **User testing – you!**

- Had submitted plan for completing computing upgrade to DOE

- Comprehensive and very successful DOE review of PDG in September 2008

- **Vital role of PDG reaffirmed**

- “The PDG publications are crucial to the field ...” (DOE reviewer)

- **DOE agreed to plan and asked us to increase request for resources for computing upgrade to ensure we will succeed**

- Now 2 FTE for 3 years (until end of FY11)
 - 0.5 FTE for ongoing support after initial development

- NSF agreed to contribute to computing upgrade according to its overall share of PDG funding

Written in 2006

High-Level Requirements and Roadmap for PDG Computing

Juerg Beringer
Particle Data Group
Lawrence Berkeley National Laboratory

This document summarizes the high-level requirements for the upgraded PDG computing system and proposes a roadmap for completing the upgrade. It is intended to serve as a starting point for a cost estimate for the completion of the upgrade project.

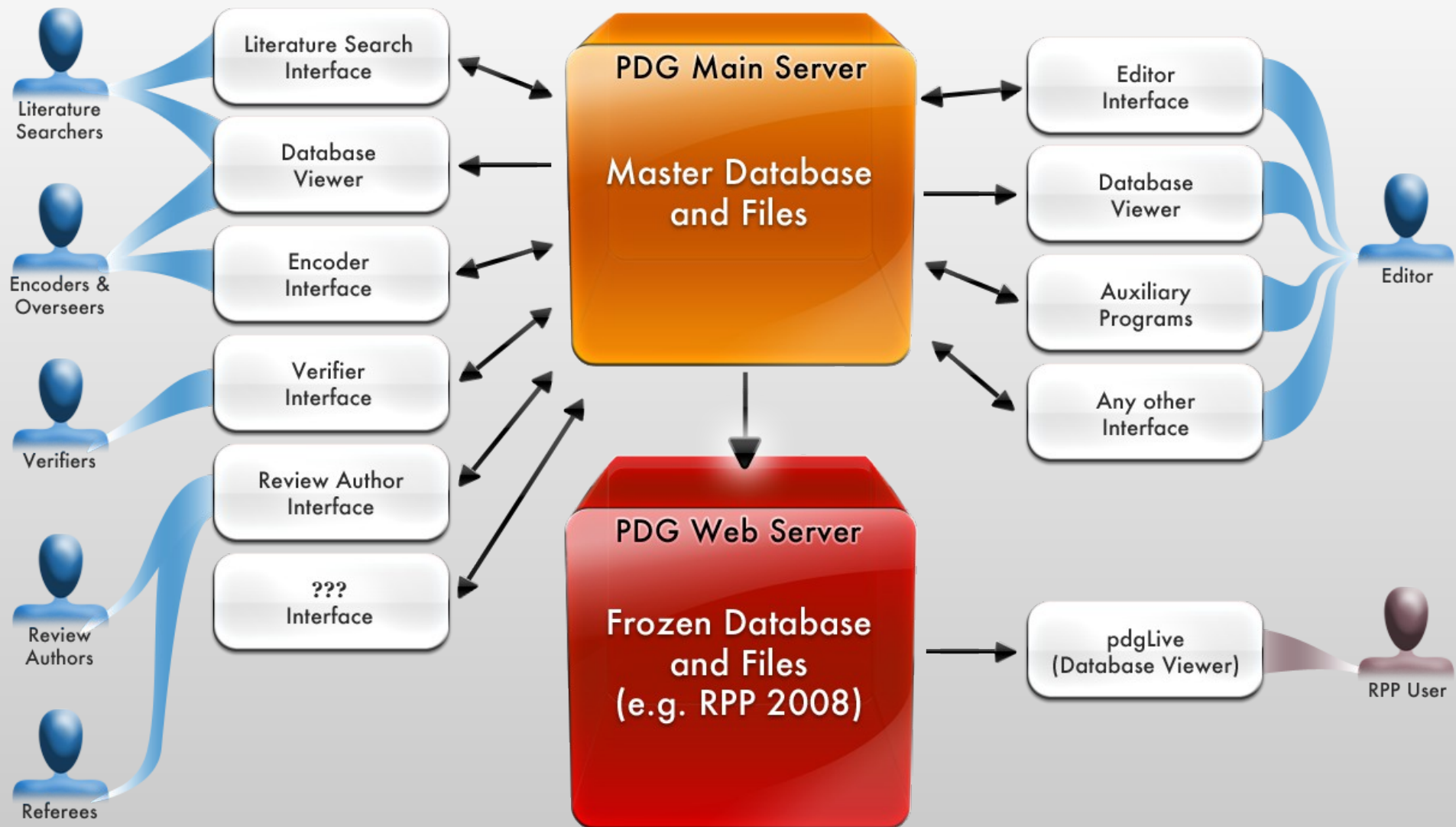
Development work funded and in full swing by end of 2008

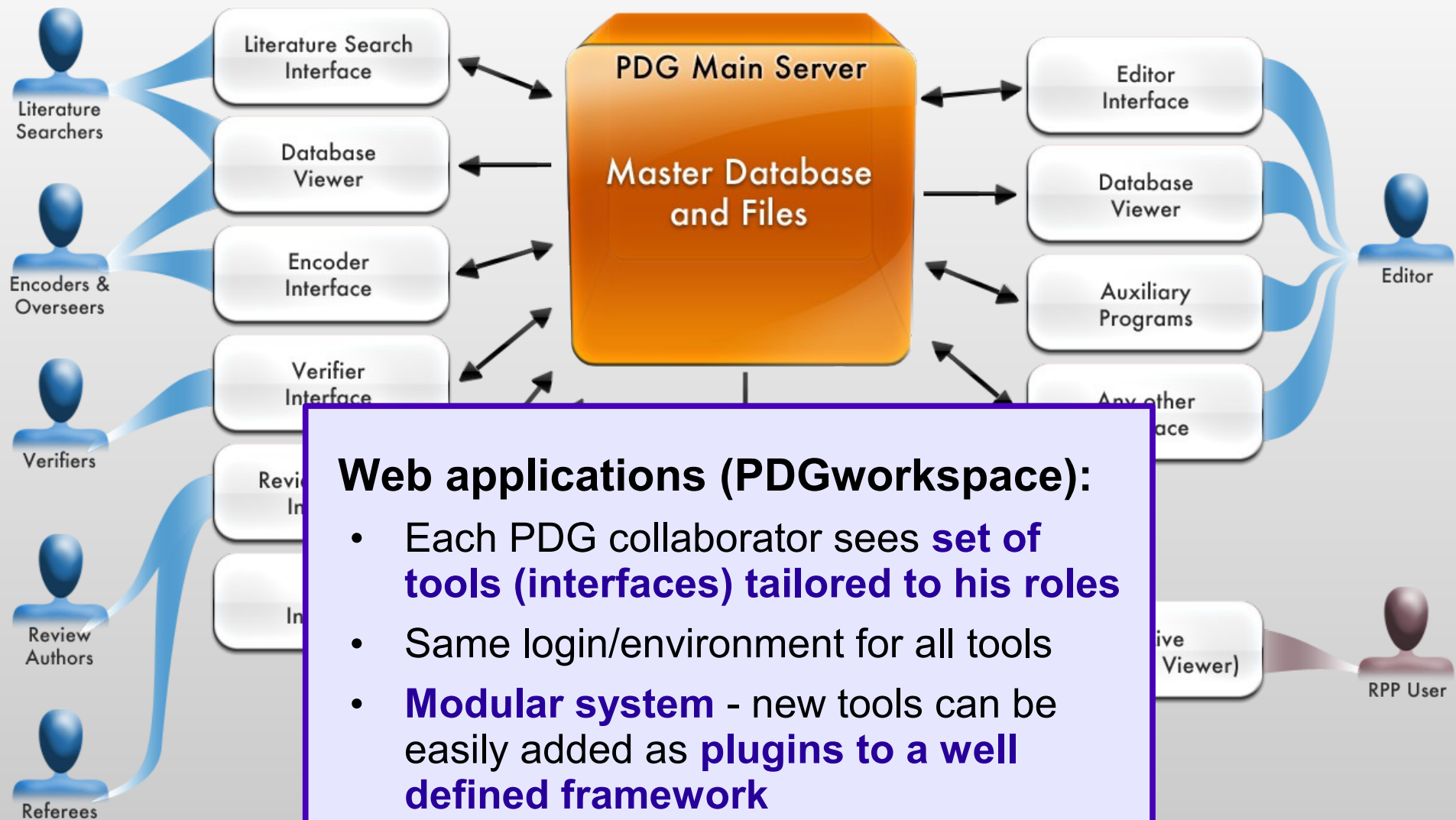
- **About two-thirds of the project completed**
- **Successfully deployed initial version (V0 Release) of new system on August 11, 2010**
 - Now our production system for ongoing PDG work
- **Full-day DOE review of Computing Upgrade on September 17**
 - <http://pdg.lbl.gov/computingreview2010/index.html>

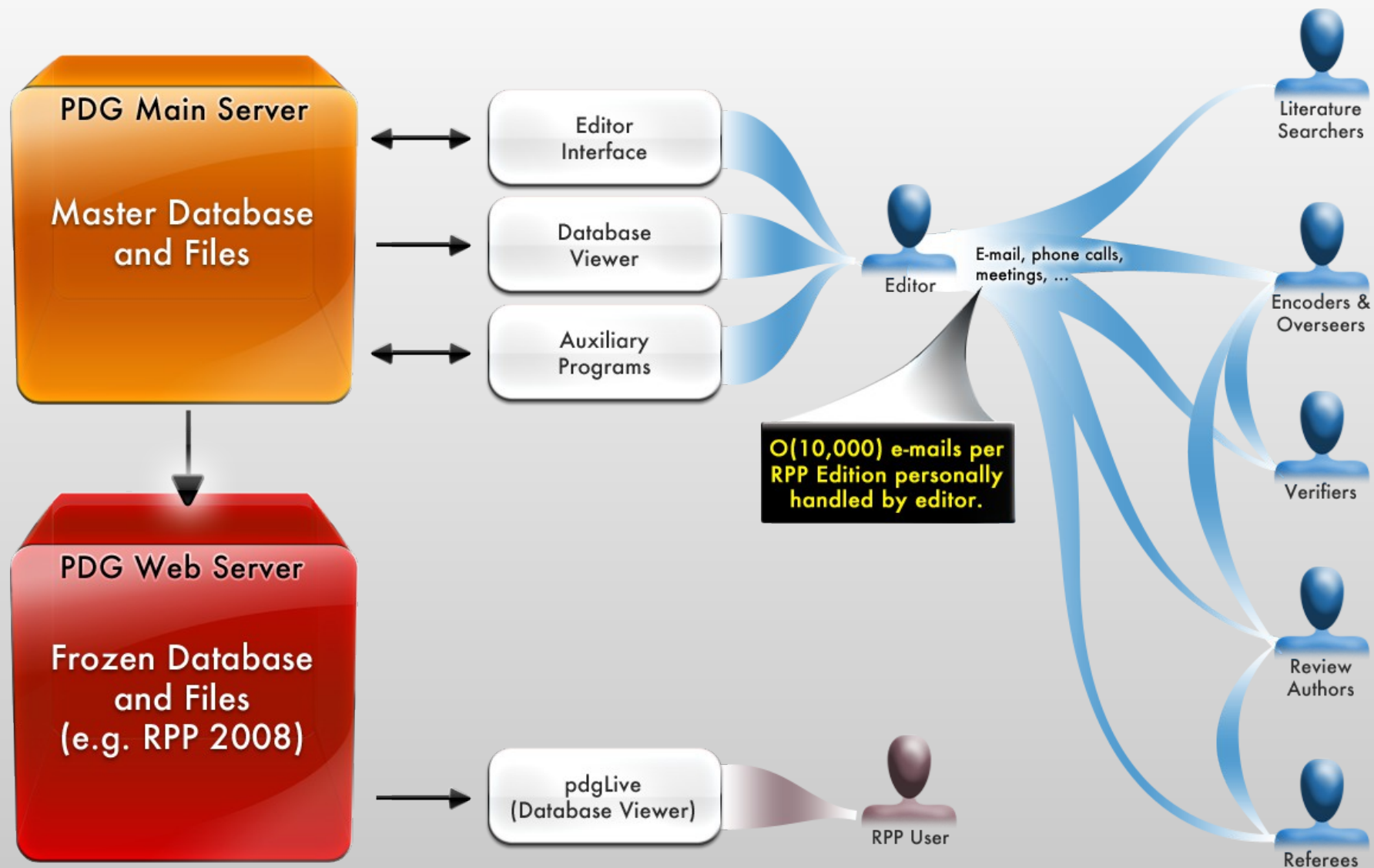
Quotes from the review report (October 20, 2010):

- **Computer system upgrade is proceeding on schedule and within cost**
- **Satisfying all the requirements laid out for it in the 2008 review**
- **Upgraded system will clearly make a dramatic improvement in the PDG production workflow**

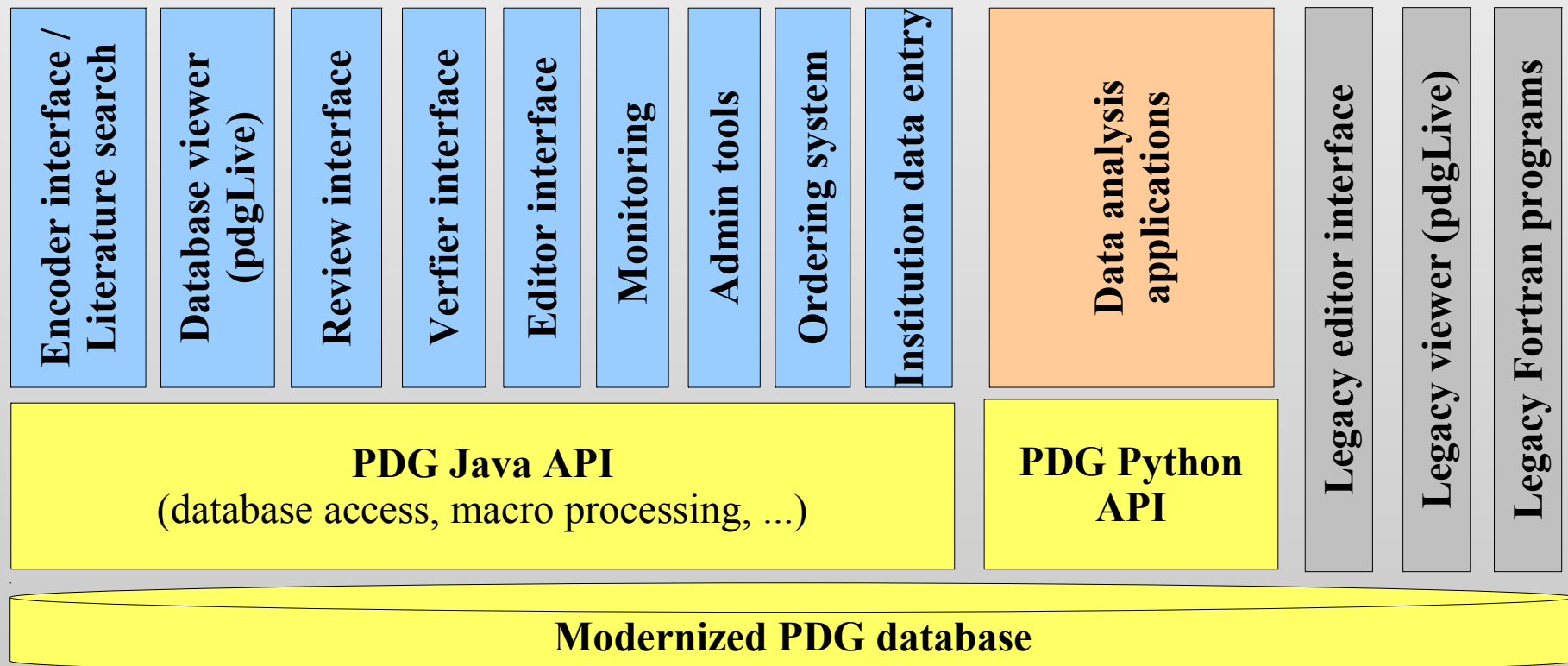
- **Primary goal: ensure that PDG can continue to function well**
 - System must be modern, modular, extendable, easy-to-use, maintainable and well-documented
- **Computing system must support all areas of our work**
 - **Decentralized, web-based data entry** and verification for Listings
 - Tools for authoring and refereeing **reviews**
 - **Monitoring** of progress in RPP production
 - Programs for evaluation of data (**fits, averages, plots, ...**)
 - **Expert tools for editor**, including creation of book manuscript and static web pages (PDF files)
 - Interactive browsing of PDG database similar to **pdgLive**
- **Suitable platform for future extensions**







- = Web applications for collaborators (PDGworkspace) and the public (pdgLive)
- = Data analysis applications
- = Legacy applications (to be phased out eventually)
- = Infrastructure (APIs, database)



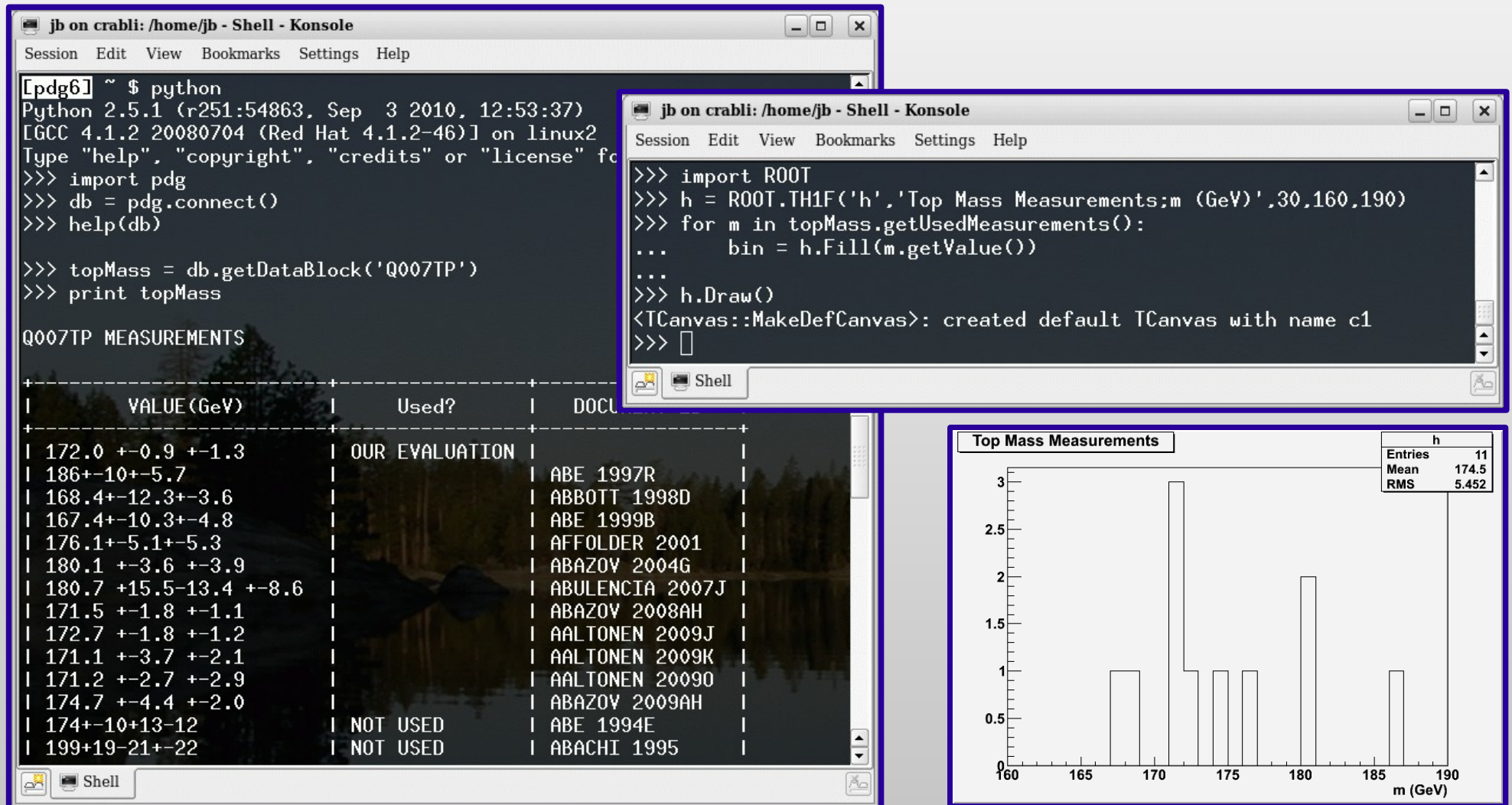
- Represents backbone of new system, including
 - Modernized PDG database
 - Java and Python APIs
 - All technologies included and working together
- All (updated) legacy applications run in V0 Release system
 - Full **production system** – now used for ongoing PDG work
- Provides modular framework into which applications can be easily and **incrementally** included (during ongoing PDG work)
- Includes **alpha release of encoding interface**
 - By far most difficult and complex application
 - Includes main building blocks required by other applications
 - Supports complete standard encoding cycle plus advanced tools

Successfully deployed August 11, 2010

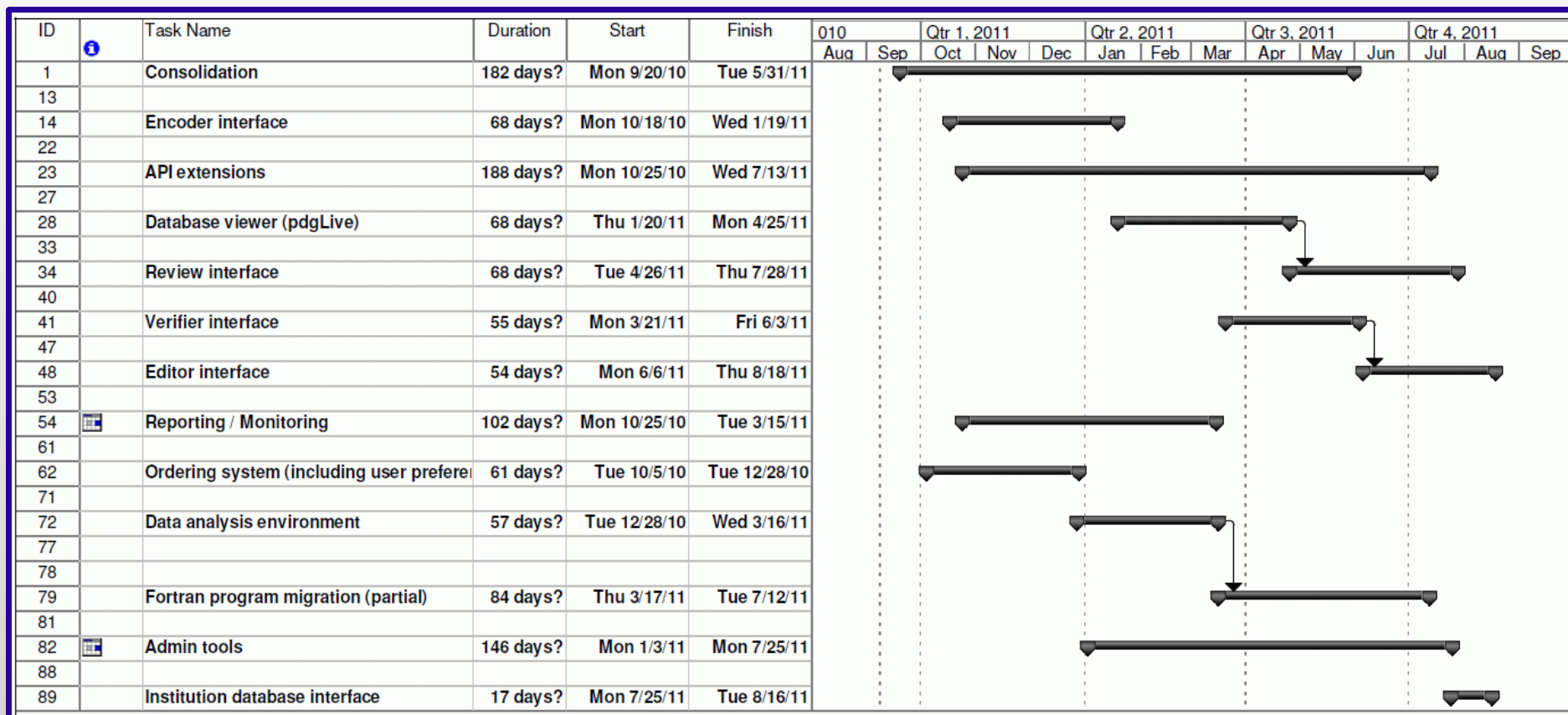
- **Encoding interface**

You'll be able to try this out yourself in a few minutes

- **Interactive access to PDG database in Python**
 - For now primarily aimed at PDG-internal use, but programmatic user access to PDG database will open whole new world of possibilities



Future Plan (Summary)

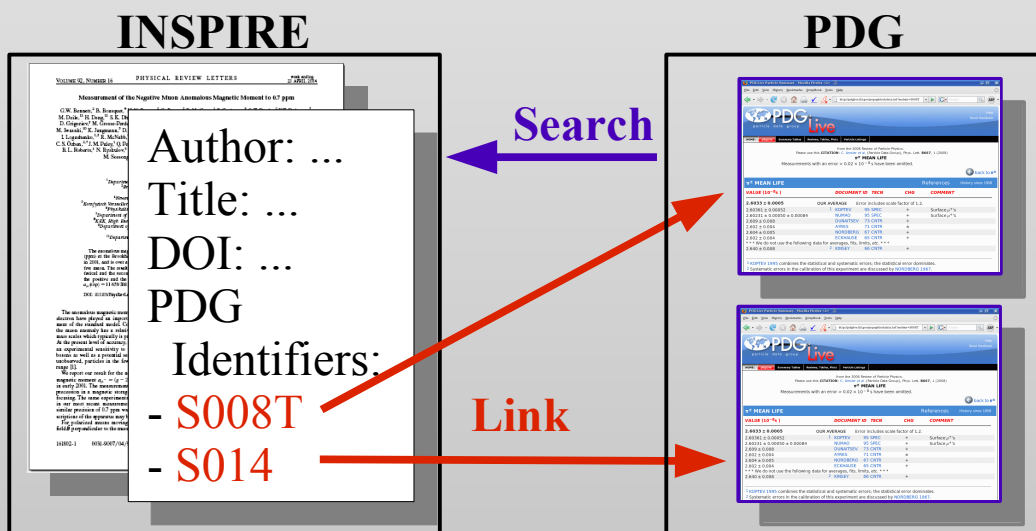


Project completion expected mid August 2011

- Leaves contingency of 3 months of funding (at present level of development effort)

- **Immediate and primary goal is to ensure PDG can continue to function well**
 - This has absolute priority over any fancy extensions
- **New computing system also provides platform where innovative new features can be implemented**
- **Several activities already started in this context:**
 - Collaboration with **INSPIRE** on cross-linking using **PDG Identifiers**
 - Participation in **HEP Information Resource Summits**
 - Presentation at CHEP'2010
 - Brain-storming about new features, e.g.
 - pdgLive on smart phones
 - Opening PDG platform to support averaging groups (e.g. HFAG)
 - User tagging of PDG content
 - Allowing programmatic user access to PDG database
 - Providing all PDG data in computer-readable form
 - ...

- **Wish list:**
 - From INSPIRE: *“What data does PDG have about this?”*
 - From PDG: *“What are the latest papers on this topic?”*
- **Permanent reference to PDG data items: PDG Identifiers**
 - Essentially PDG nodes (e.g. “Q007TP”)
 - PDG will publish authoritative list
- **Map to other classification schemes or use as “pointers”**



- **Generate initial set of tags from PDG database**
- **Could allow authors to tag their articles**

- Thanks to funding received after very successful 2008 DOE Review of PDG, the planned computing upgrade is under way and expected **to be completed around mid-August 2011**
 - “... is proceeding on schedule and within cost” (DOE review 2010)
- An initial version of the new system has been successfully deployed and is now the PDG production system
 - First release of modern, **extendable** and **maintainable** PDG system
 - Represents complete backbone of new system
 - Provides modular framework into which applications can be easily and **incrementally** included (during ongoing PDG work)
- Remainder of project devoted to building other user interfaces
 - Main **building blocks needed already available** and working in encoding interface
- Now have platform for implementing **innovative new features**
 - First example will be greatly improved cross-linking with INSPIRE

Testing the Alpha Release of the New Encoding System

- **This is still an alpha release**
 - Some things don't display correctly
 - Some things don't work or are not yet implemented
 - Some things are too slow
 - Paper assignments as done by meson team are not yet supported
 - **But you can do most encoding tasks**
 - **You can even define new decay modes and branching ratios**
- **Please try it and give us feedback**
 - Do you like it?
 - Is it sufficiently intuitive to be usable without training?
 - What can we improve?
 - Please fill out the feedback sheet

Sarah, Chuck, David, Orin, Piotr and myself are here to help (if necessary) and to listen to your feedback

- **The system manages the encoding work flow by assigning each paper to the person who is responsible for doing the next step in the encoding process**
 - When you sign off your part, the paper is assigned to the next person
 - Side remark: If several persons are eligible to e.g. encode a paper, any one of them will be able to assign it to themselves (not yet implemented)
- **When you log in as an encoder or overseer (for the Listings) you'll see a list of tasks (=papers) that you need to work on**
 - You can select to see past or future tasks
- **Try adding a measurement to an existing datablock**
- **Try adding a new datablock for a new branching ratio using the toolbox (first define a new decay mode)**
- **Any changes you make go directly into the PDG database**
- **Today we are using a demo database, so please try out freely!**

- To log into PdgWorkspace go to

<http://pdgprod.lbl.gov:8080/PdgWorkspace>

- Use your login name if listed in the table below
 - You should see some of your papers to encode for the 2011 update
 - If you're an overseer, select “Future Tasks” instead of “Pending Tasks”

cjslin	groom	masoni	rmbarnett
dambrosio	gurtu	navas	terning
eidelman	hikasa	olive	venanzoni
goodman	jelys	patrignani	wmyao
grab	jfarguin	raffelt	wohl

- Or log in as “hagiwara” to try working on top quark papers
- The password for all accounts will be announced in the session

Backup Slides

- **Not yet implemented**
- **Will also be part of PdgWorkspace**
- **Planned features include:**
 - Workflow management (authoring, refereeing; notifications)
 - Two working models supported:
 - Edit review in web-based editor (no need for local TeX/TeXsis)
 - Check out review sources, work locally, check in modified versions
 - Convenient access to different versions (including differences)
 - Consistency checking of related reviews and Listings
 - Status reporting
 - Mailing lists for each review with automatic subscription of review authors and/or referees

- **Juerg Beringer (PDG physicist)**
 - Project leader, requirements, system architecture
- **Chuck McParland (computer scientist)**
 - Java API
- **Sarah Poon (computer systems engineer)**
 - Web design, user interfaces, JavaScript
- **David Robertson (computer systems engineer)**
 - Database, Python API, scripts
- **Orin Dahl (PDG physicist, retired)**
 - Legacy Fortran programs
- **Piotr Zyla (PDG editor)**
- **Contributions from Jacob Andreas, Cecilia Aragon, Keith Beattie, Igor Gaponenko, Keith Jackson, Kirill Lugovsky, Slava Lugovsky**

Each member of the team has many years of software development experience

- **J2EE-based web application framework**
 - Commonly used industry standard for scalable, distributed web apps
- **Ajax-enabled web pages**
 - User-friendly and highly interactive GUI behavior
- **Relational database (PostgreSQL)**
 - Currently 130 database tables
 - ORM tools (Object-Relational Mapping): Hibernate, JPA; SQLAlchemy
- **Programming languages**
 - Java and JSP for web application framework backend
 - JavaScript and CSS for client-side HTML (Ajax)
 - Python API for programmatic access to database and to interface to numerical libraries and tools
 - Legacy Fortran applications restructured as libraries
- **TeX (print, PDF) and MathML (web)**

- **PDG has special requirements that cannot be addressed by “commodity software”**

Solution:



- Identified challenging areas posing potential risk to project
- Carefully addressed these areas **first** (through **design**, **technology choices**, and **project planning**)

- **Computing upgrade must proceed in parallel to PDG work**
 - Legacy system must continue to run during development
 - Severely limits opportunities for system deployment (once per year)
 - Workload on PDG experts from having to work with two systems

Solution:

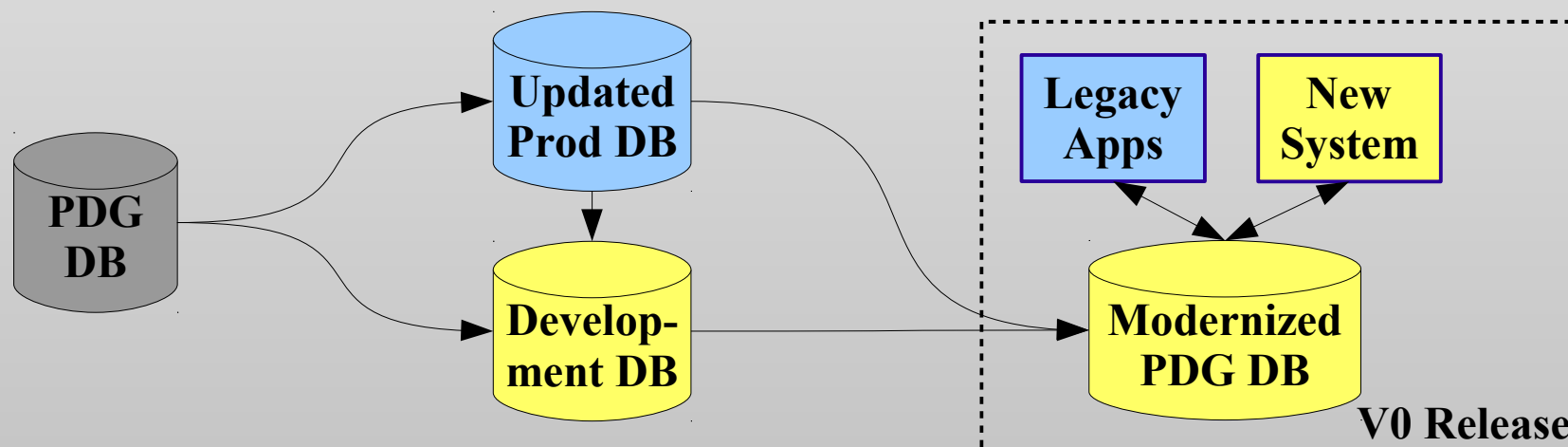


- Must carefully plan new system deployment
- Release as early as possible with legacy applications running within new system (“V0 Release”, see later)
- Allows incremental deployment of new components

- Existing scientific data must be migrated to new system
 - Complete redesign of PDG database from scratch impractical from many points of view
 - Changes to PDG database must be made incrementally
 - Small database changes mandated by ongoing PDG work
 - Conventions on how data is stored in the database (macros, flags, etc)
 - Occasionally need new columns in tables

Solution:

- Modernized PDG database used by both (updated) legacy applications and the new system



- **Scientific output from old and new system must be identical; PDG data must be correct**
 - Inherently difficult to validate tens of thousands of numbers

Solution:



- Nightly builds with unit tests
- Careful and detailed validation before use for PDG production
- Detailed logging of changes at database level
- Version control of database contents by dumping to CVS
- **System validation by producing TeX manuscript of full Review in old and new system, then making sure all changes (“diff”) are expected and desired**

- **Distributed data entry**
 - System must take care of complicated distributed work flow
 - Detailed logging of changes (“Why did this number change?”)

Solution:



- Careful design
- Suitable industry-standard technology choices (J2EE)
- Innovative logging scheme using database triggers that keeps track of logical operations and enforces logging at database level for any application (doesn't need any application specific logging support)

- **Use of TeX and display of math on the web**

Solution:



- Evaluate existing solutions (MathML, jsMath, mimeTex, TeX-to-MathML translators, ...)
- Found solution that addresses our needs (see Sarah's talk)

- **Browser and platform diversity among large user base**




Solution:

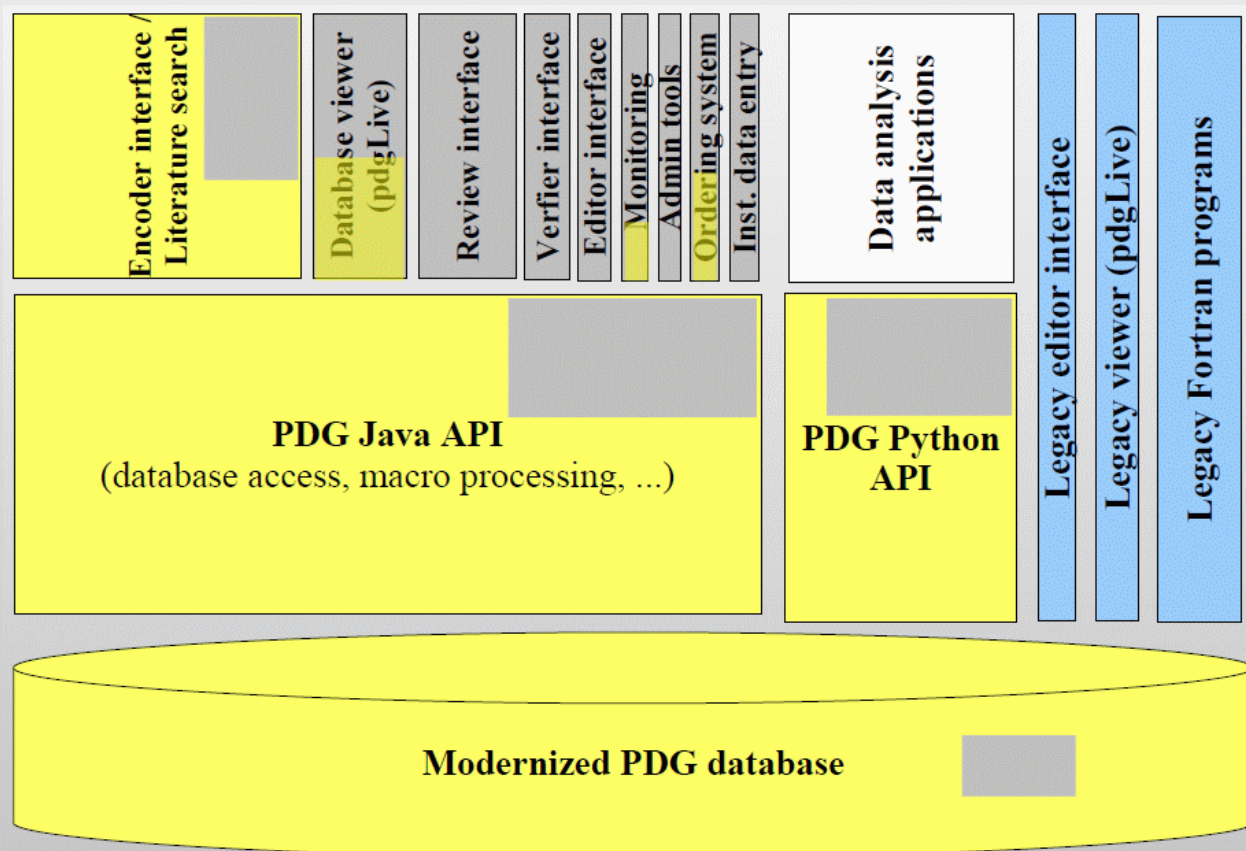


- Use existing extensive JavaScript library where this problem is already solved (see Sarah's talk)

- **Follows widely-adopted practices, including**
 - Iterative design process with close interaction with users
 - Ongoing documentation (Wiki, within code, formal manuals)
 - Nightly builds and nightly unit tests
 - Using existing tools, components and libraries to maximize efficiency
- **Frequent communication**
 - Weekly general meetings
 - Weekly individual meetings of developers with project leader
 - Additional meetings as needed
 - Mailing list
- **Close involvement of PDG members**
 - So far through Orin, Piotr and myself (plus occasionally Cheng-Ju Lin and Weiming Yao)
 - As user testing ramps up, will increasingly involve other members of LBNL PDG group plus selected members from PDG collaboration

- Rescaled diagram to reflect approximate development effort

-  = updated legacy applications (in V0 release)
-  = new components included in V0 release
-  = still to be implemented as part of upgrade (some partly done)



- Initial design and planning ✓
- System architecture ✓
- Database abstraction layer ✓
- Encoder interface and literature search interface **mostly** ✓
- Database viewer **(main building blocks available)**
- Data analysis environment **partly** ✓
- Review interface
- Other system tasks
 - Refactor existing auxiliary programs ✓
 - Status monitoring
 - System monitoring **partly** ✓
 - Verifier interface
 - Editor interface
 - Ordering system **partly** ✓
 - Institution data entry
- Final acceptance test

- Initial design and planning ✓
- System architecture ✓
- Database abstraction layer ✓
- Encoder interface and literature search interface **mostly** ✓
- Databases
- Data analysis
- Review interface
- Other system components
 - Refactoring
 - Status monitoring
 - System integration
 - Verification
 - Editor
 - Ordering
 - Institution data entry
- Final acceptance test

• All difficult parts posing potential risk to the project are implemented

• The encoder interface is by far the most complex and difficult application to implement

• The encoder interface includes the building blocks needed for the other applications (e.g. macro processing, math display, etc)

• Therefore, building the remaining applications will be relatively fast

To give an approximate measure of the size of the source code developed, here are some numbers of lines of source code:

- **Java API** **75k**
 - Related to database (of which 38k generated) 44k
 - Related to macro processing 22k
 - Related to unit tests 9k
- **Encoder interface** **16k**
 - Java 8k
 - CSS 2k
 - HTML, JSP, JavaScript 6k
- **Python API** **1k**
- **Migration scripts (SQL, some Python)** **3k**
- **Legacy Fortran programs (incl. 45K comment lines)** **110k**